A Completely Digital, Low-Power, and Low-Area Phase Synchronization Architecture for Meso-Synchronous Clock Domains Supporting Dynamic Frequency Scaling

Anurag Choudhury– Texas Instruments (India) Pvt. Ltd.
Robin Hoel – Texas Instruments (Norway) Pvt. Ltd.
Abhishek Verma – Texas Instruments (India) Pvt. Ltd.
Aniruddha PN – Texas Instruments (India) Pvt Ltd
Abhinav Parashar – Texas Instruments (India) Pvt. Ltd.

62 THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA® IEEE Council on Electronic Design Automation sigda acm

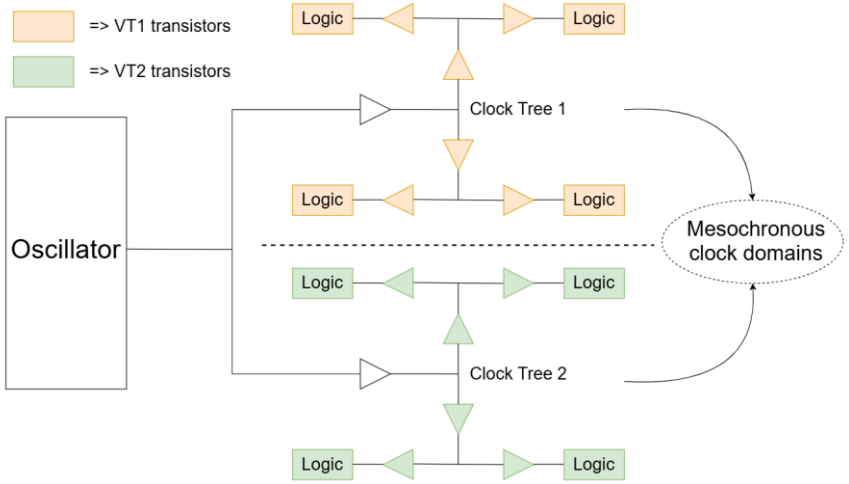
Anurag Choudhury

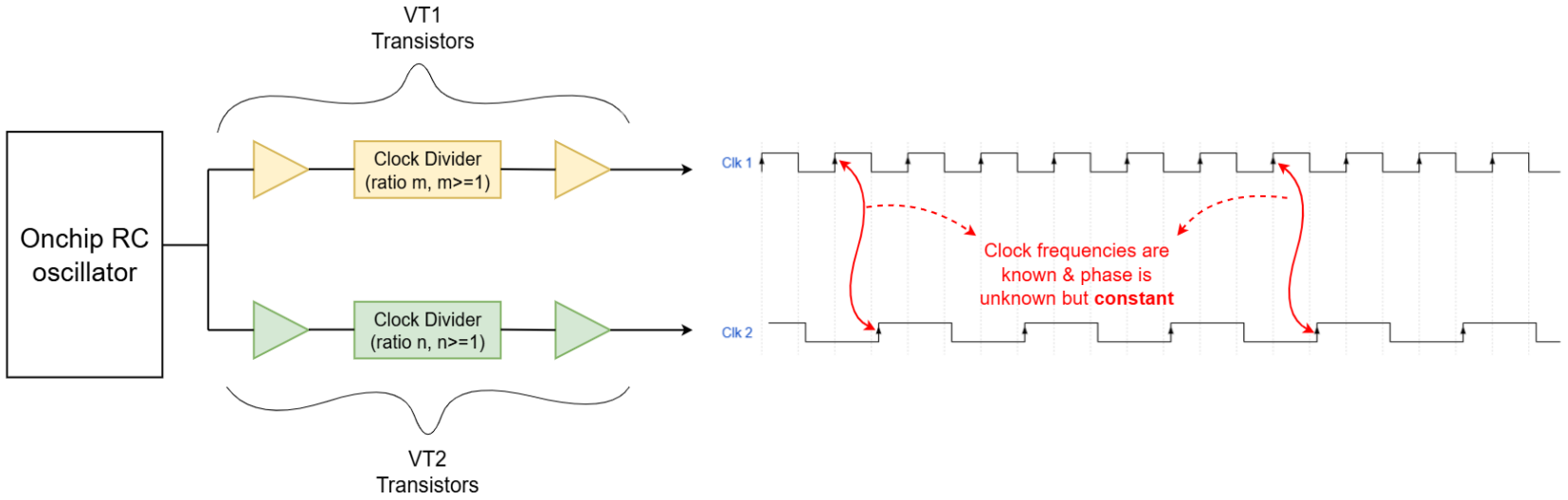
Design Engineer,
Texas Instruments India Pvt Ltd

TEXAS INSTRUMENTS

What are mesochronous clock domains?

- Two clock domains are mesochronous if the clock **frequencies** are **known** but **phase** is **unknown but constant**
- Ultra low power SOCs have multiple clock frequencies derived from a **single** oscillator to achieve the target power requirements
- More aggressive power targets necessitate that clock trees for different clocks are built using **multi-V_t transistors**, making the clocks mesochronous





Challenges and Motivation

1. Throughput and performance

- Mesochronous clock domains can be treated as purely asynchronous, but that leads to **reduction** of throughput and performance
- At least **two** clock cycles are **lost** in each direction due to synchronization

2. Architecture scalability

- The synchronization architecture should support **multiple frequencies** without needing any hardware changes
- Should cater towards **both burst** and **sporadic** data transfers
- Should **not** have **technology** or gate level **dependencies**

3. Balancing performance, power and area

- Maximum allowable performance **without** compromising on **power** and **area**

4. Existing prior art

As per existing literature, the mesochronous domain synchronizing techniques can be categorized in either of 4 architectures:-

a) **Standard Two-Clock Async FIFO**: Widely used for synchronizing different clock domains due to robustness, though it has higher area compared to mesochronous-specific solutions.

b) **Delay-Line Based Synchronizers**: Use adjustable delay lines to synchronize data and clock signals, but require precise delay matching and are best suited for custom designs.

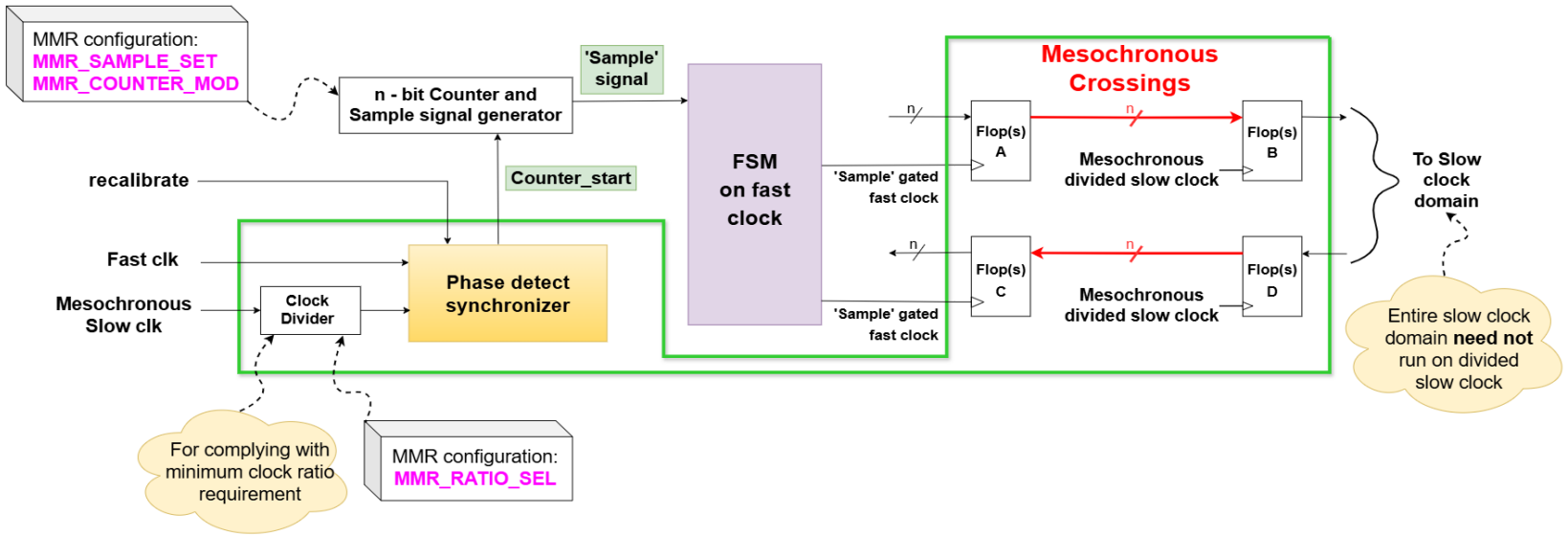
c) **Phase Detection Methods**: Predict clock phase conflicts and adjust signal timing accordingly. While effective, these designs often have higher complexity.

d) **Token-Based FIFO Systems**: These designs implement cyclic buffers to prevent metastability during data synchronization, but typically require at least 4 depth buffers

Proposed Solution

- The proposed phase synchronization scheme is shown in the figure below:-

Synchronization Scheme for Mesochronous clock domains



Behavioral RTL code of fast clock FSM:

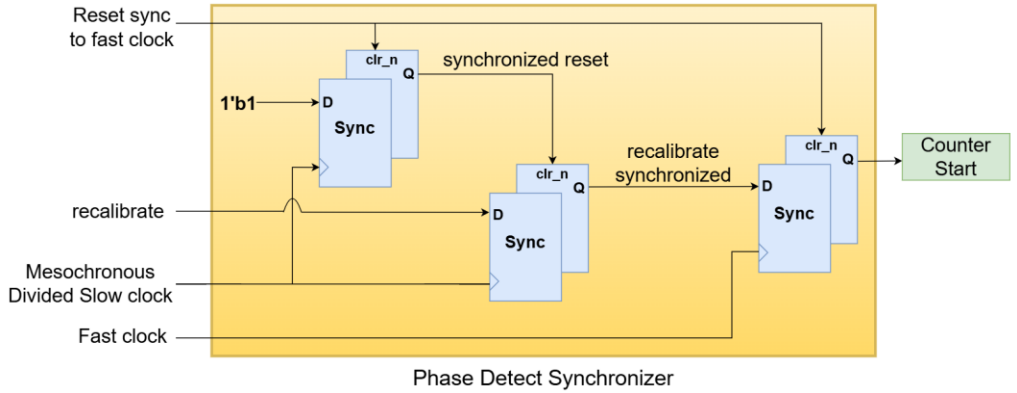
```
always @(posedge fast_clk, negedge rstn)
begin
   if (rstn) begin
      flop_A <= 'b0;
      flop_C <= 'b0;
   end
   else if (Sample) begin
      flop_A <= fast_clk_domain_data;
      flop_C <= slow_clk_domain_data;
   end
end
```

- The key principle for the working of this overall scheme is mentioned below :-
  - The flops on **fast clock** (i.e. flop(s) A & C) **launch** and **capture** data at a **safe margin** from slow clock, as directed by the '***Sample'*** signal.
  - Flops on **slow clock** (i.e. flop(s) A & C) are **free running.**
  - For the scheme to work, the ratio of fast to slow clock should be in the form of **p/q (p and q are integers).** E.g. A ratio of **2.4** can be expressed as **12/5,** hence shall be supported by this architecture proposal.

Proposed Solution

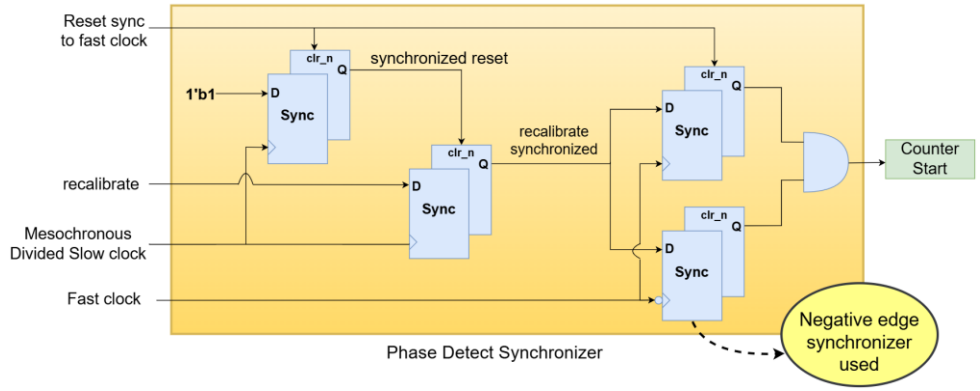
- The phase detect synchronizer is implemented completely in digital and 2 schemes are proposed for the same:

Scheme 1



- Uses **only** positive edge synchronizers for detecting phase.
- Ratio (N) between the two mesochronous clocks needs to be **greater than or equal to 4.0**

Scheme 2



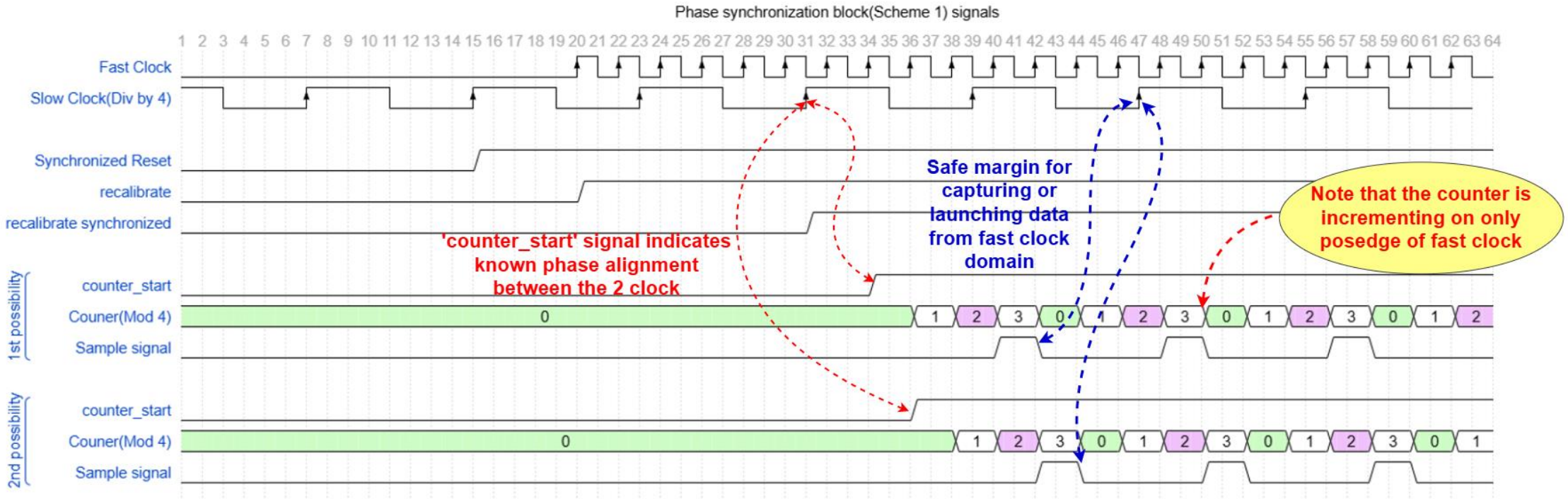
- Uses **both** positive and negative edge synchronizers for detecting phase
- Ratio (N) between the two mesochronous clocks can be **greater than or equal to 2.0**

Phase detect synchronizer operation:

- The phase detection circuit generates a ***'counter_start'*** signal, which kickstarts a modulo N counter from a **known phase alignment** between the edges of fast & slow clock at cold boot or after device standby cycles.
- Logic inside the **green boundary** is minimal and modular ensuring that the phases of clocks input to the phase detect synchronizer, are **similar** to clock phases input to Flop(s) A, B, C and D.
- '***recalibrate***' signal can be used to restart the phase alignment optionally, during the mesochronous crossing idle periods, or when either domain undergoes a standby/reset cycle.

Proposed Solution

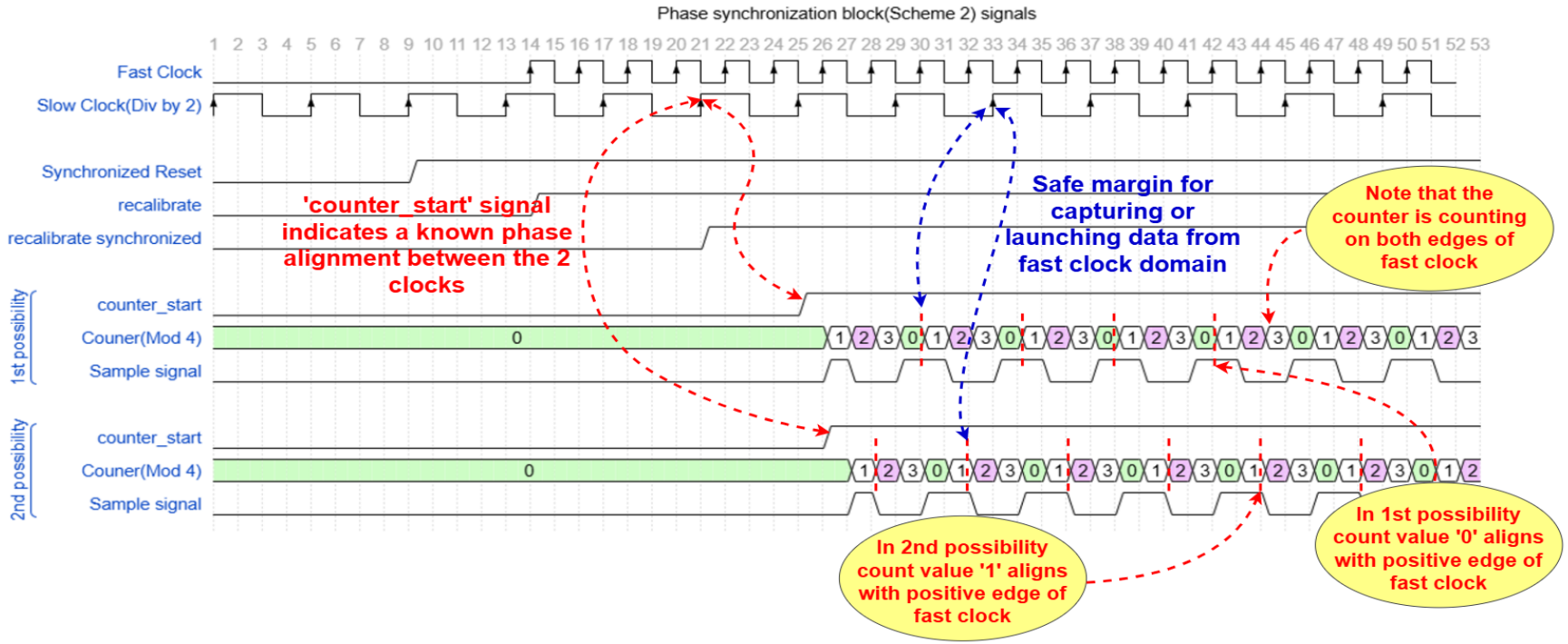
- A detailed annotated waveform of scheme 1 (with clock ratio of 4) of phase detect synchronizer is shown below:



- The *'counter_start'* signal will always go high on the 2nd or 3rd edge of Fast clock, depending on when the synchronizer settles. **Both possibilities** are displayed
- A count value of **3 (set by MMR_SAMPLE_SET)** best suited to set or capture data at a safe distance from positive edge of slow clock from the fast clock, is indicated when '*Sample'* is set, in both possibilities.

Proposed Solution

- A detailed annotated waveform of scheme 2 (with clock ratio of 2) of phase detect synchronizer is shown below:

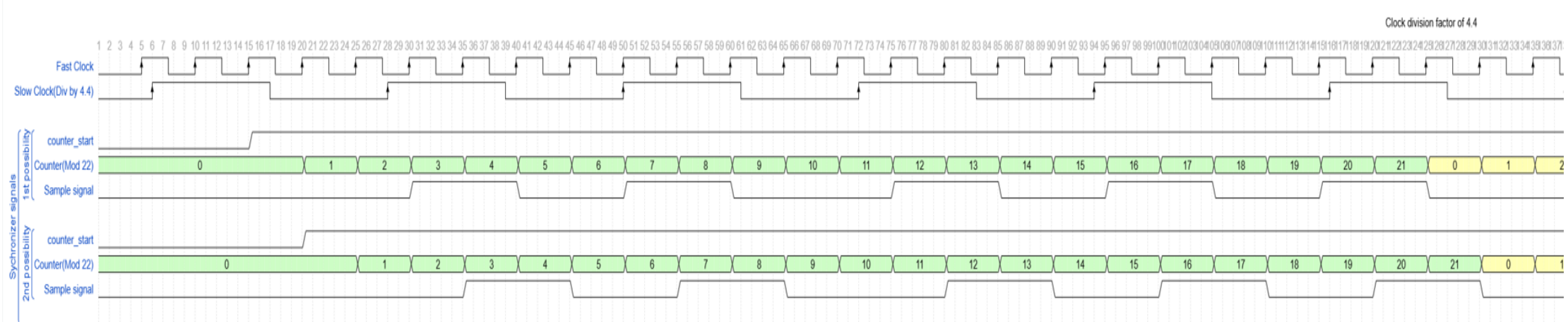


- The counter counts on **both** positive and negative edges of the fast clock after *'counter_start'* is set by the phase detect circuit.
- *'Sample'* is high for **2 half cycles** of fast clock (count value **0** and **1**), but only **one** of them is aligned with **positive** edge of fast clock. This edge is used to drive and sample data from fast clock to slow clock safely across the mesochronous domains.
- **Half cycle** paths will exist from negative edge triggered counter flops to final positive edge triggered flops in fast clock domain (i.e. Flop(s) A & C from earlier), but these do not result in timing criticality, as the logic depth on these paths is negligible.

Proposed Solution

What about non integer ratios?

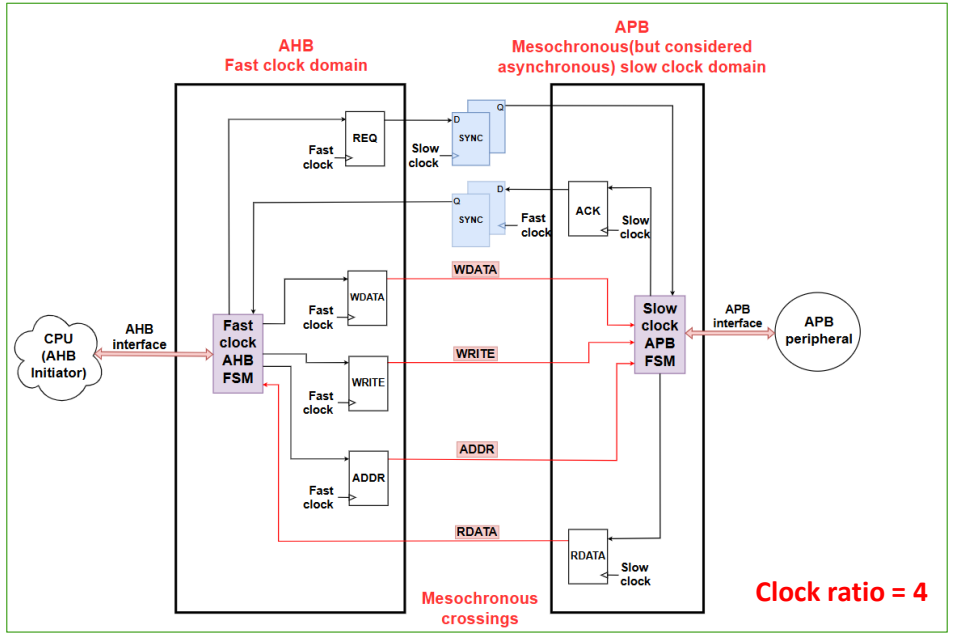
- The below waveform is counter implementation for clock division ratio of **4.4**



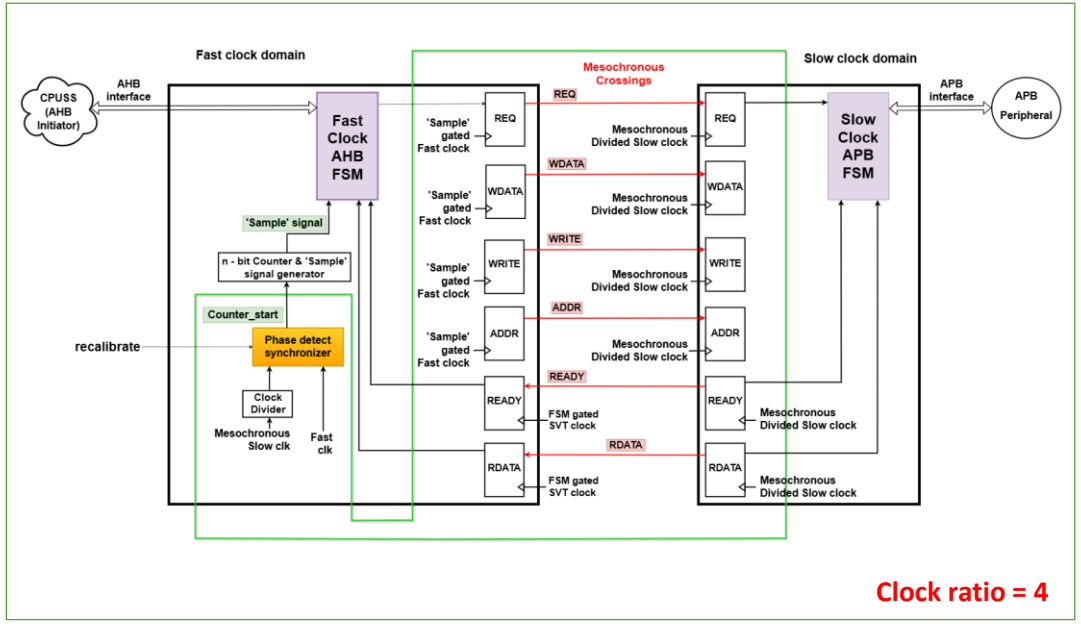
- Since the ratio of 4.4 can be represented as **22/5**, which is a ratio of 2 integers, the scheme holds good
- We run a **Mod 22** counter for a clock ratio of 4.4

Result

- The proposed phase detect synchronization scheme has been used to implement an AHB to APB mesochronous bridge, which was traditionally implemented using 2 flop synchronization scheme



Vs



Previous architecture: Mesochronous domains considered as **asynchronous**. Each transaction involves synchronization **delay** of *two fast clocks + two slow clocks*

Proposed architecture of phase synchronization for mesochronous clock domains. Once phase difference is identified, **no** additional sync delay is introduced for subsequent transfers, and data can be sampled and received at **every slow clock** edge henceforth.

Result

Results (Implemented on 65nm tech. node, Area: 1.61kgates, Throughput increase: 275%):

Results	Previous architecture (Considering mesochronous domain as async)	Other known pure digital architectures for mesochronous domains[1][2][3][4]	Proposed architecture
Throughput	1x (18-23 cycles/transaction)	Up to 2.75x	2.75x (8 cycles/transaction)
Area	1x (~105 flop equivalent, 1.02kgates)	Minimum 9x-12x	1.6x (~162 flop eq., 1.61kgates)
Dynamic frequency scaling	Supported	Not Supported / Area intensive solution	Supported

Metrics Compared with Prior Art

Metrics	Async clock domains	Existing literature[1][2] [3][4]	Proposed architecture
Area	1x	9x-12x for pure digital implementation	1.1x – 1.6x
Performance	1x	Up to 2x	2x
Technology dependency	No	Not for pure digital, Yes for analog solutions	No
Targeted use case	Sporadic and burst data	Burst data. First data of the burst incurs delay.	All - Sporadic and burst data
Dynamic frequency scaling/Multiple frequency support	Supported	Not Supported or Area intensive solution	Supported (by configurable MMRs)
Minimum frequency ratio requirement	None	None	2.0 (with posedge and negedge edge flops) / 4.0 (with only positive edge flops)

Summary

- Proposed phase synchronization architecture is extremely **area-efficient** adding only:
  - Digital phase detect circuit built by 3-4 digital synchronizers for synchronous reset
  - Modulo N counter and '***Sample***' signal generator (~log2(N)+2 flops)
  - An FSM to push only '***Sample***' gated fast clocks for driving & sampling data to/from mesochronous slow clock domain (<10 flops)

- It is **power-efficient**, since the phase detect circuit, used to trigger the '***counter_start***' is enabled very **sporadically:**
  - At **cold boot**
  - After **standby/reset cycles** where one domain is turned off and on
  - After **long** periods of **idle** time, where the application CPU can choose to recalibrate and restart the modulo-N-counter.

  The double synchronizer structures and the phase detect circuit are **clock gated** to save power.

- **All** mesochronous crossings of an SOC can be handled by a **single** phase detect synchronizer circuit, thus providing **highly scalable area saving** along with **highest possible throughput** ensuring that no slow clock or fast clock cycles are spent for synchronizing data

- Scalable architecture, compatible with **any clock ratio**, which is greater than minimum specified requirement